FACULTY WORKING PAPER 93-0120

Input Control in Job Shops

Narayan Raman
Department of Business Administration
University of Illinois

Bureau of Economic and Business Research
College of Commerce and Business Administration
University of Illinois at Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 93-0120

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

March 1993

Input Control in Job Shops

Narayan Raman
Department of Business Administration

Input Control in Job Shops

Narayan Raman

Department of Business Administration
University of Illinois at Urbana-Champaign
Champaign, Illinois 61820

September 1991
Revised March 1993

ABSTRACT

Input Control plays a critical role in regulating the release of jobs into a production system, and thereby, controlling the inventory levels. For a just-in-time system, we model the input control decision as a bicriterion problem which requires the maximization of the sum of job release times subject to minimum total job tardiness.

We propose a solution method that requires partitioning the set of available jobs into blocks; jobs in the same block are processed without any inserted idle time. The suggested procedure updates the membership of each block, determines the schedule of jobs within each block as well as the schedule of blocks in an iterative manner

by solving a sequence of subproblems. Computational results show the effectiveness of the proposed method as well as the substantial improvement that can be affected in job release times, without adversely affecting the tardiness values, over an approach that considers the single objective of minimizing total tardiness. This result is of interest to an operating manager who is responsible for controlling work-in-process and finished goods inventory related costs in addition to meeting due dates effectively.

1 Introduction

In many manufacturing systems, the release of jobs into the production shop floor is regulated by means of an input control mechanism. The purpose of this mechanism is to allow jobs to be introduced selectively into the system instead of releasing them whenever they become available. In most manufacturing systems, this leads to important savings. First, it reduces work-in-process levels that results in less congestion on the shop floor. Second, it simplifies operation scheduling and smooths system output. Third, input control promotes just-in-time (JIT) manufacture in that it discourages early job completions. In so doing, it improves the robustness of the master production schedule because, by deferring jobs required at a future date, the system is less susceptible to demand and due date variations on account of factors such as order cancellations. Input control is also imperative in many flexible manufacturing systems (FMSs) because of the limited buffer space available. Stecke (1983), for example, proposes a hierarchical approach to FMS planning and scheduling in which the determination of part input sequence precedes the operations assignment and tool loading decisions. Other instances when input control is required include shop load leveling (Irastorza and Deanne 1974) and load balancing (Shimoyashiro et al. 1984).

While there are many practical benefits of using input control, Baker (1984a) notes its important drawbacks as well. By removing some of the options available to the scheduler, input control may be counterproductive. In particular, if the scheduling objectives are *regular* measures, i.e., if the schedule costs are nondecreasing in job completion times, then delaying job input could easily result in suboptimal solutions. However, the success of JIT manufacture has highlighted the costs incurred when jobs are completed well ahead of their due dates. This has generated increased interest in nonregular scheduling measures, such as minimizing earliness, that promote the notion that jobs should be released as close to their due dates as possible. Therefore, as Baker and Scudder (1990) note, due date considerations under JIT production require addressing both tardiness and earliness related costs. The purpose of input control in such systems is to determine the job release times that minimize these costs.

The earliness of any job is usually defined as the positive difference between its due date

and its completion time. However, if the job comprises multiple operations to be done on different machines, the cost of completing the job ahead of its due date should also consider the waiting times between operations. In so doing, the input sequence accounts for the cost of work-in-process inventories in addition to the finished goods inventories. As we discuss in §2, minimizing the sum of earliness and the total waiting time for any job is equivalent to maximizing the time it is released into the system. Furthermore, by insuring that the raw materials required for any job arrive at the time the job is scheduled to start its first operation as determined by the input sequence, raw material inventories can be largely avoided. Therefore, from a scheduling perspective, maximizing job release times appears to be an appropriate surrogate for minimizing inventories. [Also see Ahmadi and Bagchi 1992 for a related discussion.]

In this paper, we consider the *static* input control problem in a job shop. For the given set of available jobs with individual due dates and processing requirements, we model this decision as the problem of maximizing the sum of job release times subject to the minimization of the total job tardiness. Thus, we consider the tardiness and earliness costs in a hierarchical manner. This objective is consistent with the results of surveys of industrial scheduling practices conducted by Panwalker, Dudek and Smith (1973) and more recently by Smith et al. (1986). These surveys find that operating managers consider meeting due dates as their most important objective, and other scheduling criteria are considered only after the best schedule for meeting job due dates has been determined. We consider minimizing total tardiness as the objective that captures the importance of meeting due dates. In a just-in-time system with its emphasis on minimizing inventories, it is natural to consider maximizing the sum of job release times as the secondary objective.

Previous research on combining earliness and tardiness measures has primarily considered the objective of minimizing their weighted sum in single machine systems. Baker and Scudder (1990) mention the paucity of work done for the case in which jobs have distinct, prespecified due dates. Fry, Armstrong and Blackstone (1987), Garey, Tarjan and Wilfong (1988), Kim and Yano (1986), and Yano and Kim (1991) give procedures for finding the optimal job start times when the job sequence is known. Fry et al. (1987) give a pairwise-interchange based

heuristic procedure for finding this sequence when the earliness and the tardiness weights are the same across all jobs. Kim and Yano (1986) give a branch and bound procedure for the unweighted case, and Yano and Kim (1991) present a branch and bound procedure as well as several heuristic methods when the earliness and the tardiness weights are distinct for each job.

However, the literature on the earliness and tardiness measures, considered individually is quite rich. Because of its NP-completeness (see, for example, Rinnooy Kan 1976), prior research on the job shop tardiness problem has primarily considered the use of priority dispatching rules. Baker (1984b) and Vepsalainen and Morton (1987) give extensive discussions on this research. In a recent study, Raman, Talbot and Rachamadugu (1989) develop an implicit enumeration approach as well as a decomposition based heuristic for solving this problem.

While the objective of maximizing the sum of job release times in a job shop has not yet been addressed directly, there is considerable literature on this subject for single machine systems for which this objective reduces to that of minimizing earliness. Baker and Scudder's survey (1990) provides a detailed description of the work done on this objective and its variants. Ahmadi and Bagchi (1992) extend this work to a two-machine flow shop for the objective of minimizing total job idleness.

Thus, this study differs from the previous research dealing with the simultaneous minimization of tardiness and earliness in two aspects. First, we treat these two costs lexicographically. Second, we extend the previous work that addressed single machine systems to a job shop. This paper is organized as follows. In §2, we give a mathematical programming formulation of the input control problem. In §3, we discuss a possible decomposition of this problem, and bring out the various embedded subproblems. The proposed solution procedure is given in §4, and our computational experience with this approach is described in §5. We conclude in §6 with a summary discussion.

2 Problem Statement

We consider a static problem with N jobs that are all available for scheduling at time zero in an M-machine job shop. Let $\mathcal{J} = \{1, \ldots, N\}$ denote the set of jobs. Job $j \in \mathcal{J}$ comprises N_j operations that need to be done in a given order on prespecified machines; however, the machine visitation sequence varies from one job to another. Let O_{ji} denote operation i in job j. The input parameters include the operation processing time p_{ji} of O_{ji}, job processing time $p_j = \sum_{i=1}^{N_j} p_{ji}$, and job due date d_j. It is then possible to determine the set $\mathcal{A}_j$ of pairs of adjacent operations in job j, $(i,l) \in \mathcal{A}_j$ if operation i is the immediate predecessor of operation l in job j. Let μ_m be the set of all operations processed on machine m. The decision variables include the job release time r_j, and the start time s_{ji} and the completion time c_{ji} of operation O_{ji}. Jobs are released into the system when they are scheduled for their first operation. Hence, $s_{j1} = r_j$, and $s_{ji} \geq c_{j,i-1}$ for $i = 2, \ldots, N_j$. Let $c_j = c_{jN_j}$ be the completion time, $T_j = max(0, c_j - d_j)$ be the tardiness, and $E_j = max(0, d_j - c_j)$ be the earliness of job j.

The job shop input control problem is stated as

ICP

$$Z_2 = \ max \ \sum_{j=1}^{N} r_j \tag{1}$$

subject to

$$Z_1 = \sum_{j=1}^{N} T_j = min_{\sigma \in \mathcal{S}}\{T(\sigma)\} \tag{2}$$

$$c_{uv} - c_{ji} \geq p_{uv} \vee c_{ji} - c_{uv} \geq p_{ji}, \ \ \forall \, O_{ji}, O_{uv} \in \mu_m, \ \ \forall m \tag{3}$$

$$c_{j1} - r_j = p_{j1}, \ \ \forall j \tag{4}$$

$$c_{jl} - c_{ji} \geq p_{jl}, \ \ \forall (i,l) \in \mathcal{A}_j and \ \ \forall j \tag{5}$$

$$c_{jN_j} + E_j - T_j = d_j, \ \ \forall j \tag{6}$$

$$c_{ji} \geq 0, \ \ \forall i, j; \ \ r_j, E_j, T_j \geq 0, \forall j \tag{7}$$

where $\mathcal{S}$ is the set of all feasible schedules for **ICP**. Equation (1) refers to the secondary objective of maximizing the sum of job release times. Constraint (2) specifies the primary

objective of minimizing total tardiness. In the given expression, $T(\cdot)$ refers to the total tardiness incurred in schedule $(\cdot)$. Constraints (3) represent the disjunction which insures that no more than one operation is processed on a machine at any time. Constraints (4) and (5), respectively, require that the first operation of any job is started when the job is released, and any subsequent operation is scheduled only after its predecessor operation is completed. Constraints (6) define tardiness while Constraints (7) specify the nature of the variables.

ICP generalizes two well-known NP-complete problems, and therefore, NP-complete itself. (2) – (7) define the job shop minimum total tardiness problem. This problem is known to be NP-complete even when $M = 1$ (Du and Leung 1990). If $Z_1 = 0$, then (1), (3) – (7) define the problem of maximizing the sum of job release times subject to all jobs being completed on or before their due dates. Let w_{ji} be the operation waiting time of job j after operation i. Then

$$c_j = r_j + p_j + \sum_{i=1}^{N_j-1} w_{ji}$$

and when all jobs are completed on or before their due dates

$$d_j = c_j + E_j = r_j + p_j + \sum_{i=1}^{N_j-1} w_{ji} + E_j.$$

Hence,

$$\sum_{j=1}^{N} r_j = \sum_{j=1}^{N} d_j - \sum_{j=1}^{N} p_j - \sum_{j=1}^{N} \left[\sum_{i=1}^{N_j-1} w_{ji} + E_j \right].$$

Because job due dates and processing times are given, maximizing the sum of job release times is equivalent to minimizing the sum of operation waiting times and earliness over all jobs. For single machine problems, $N_j = 1$, $\forall j$, and maximizing $\sum_{j\in\mathcal{J}} r_j$ is equivalent to minimizing total earliness $\sum_{j\in\mathcal{J}} E_j$. Chand and Schneeberger (1986) show that this problem is NP-complete. For given operation processing times, $c_{j1} = r_j + p_{j1}$, and maximizing $\sum_{j\in\mathcal{J}} r_j$ is equivalent to maximizing $\sum_{j\in\mathcal{J}} c_{j1}$ which is considered by Ahmadi and Bagchi (1992) for a flow shop.

3 The Notion of Critical Jobs

In general, an optimal schedule will have inserted idle times because the secondary objective is nonregular; this results in one or more job *blocks*. A block B_k satisfies the property that for any job v in B_k, there is another job u in B_k such that if $c_v \geq c_u$, then $c_u > r_v$. Figure 1 shows a schedule comprising three blocks for a 3-machine, 11-job problem. [In this figure, M1, M2 and M3 denote the individual machines.] Let $\mathcal{B}_k$ denote the set of jobs in block B_k. It is easily seen that for two adjacent blocks $[k]$ and $[k+1]$ in any sequence,

$$max_{j \in \mathcal{B}_{[k]}}\{c_j\} < min_{j \in \mathcal{B}_{[k+1]}}\{r_j\}. \tag{8}$$

INSERT FIGURE 1 HERE

For block B_k, we define the block release time R_k, block completion time C_k, and block due date D_k as:

$$R_k = min_{j \in B_k}\{r_j\} \tag{9}$$

$$C_k = max_{j \in B_k}\{c_j\} \tag{10}$$

$$D_k = min_{j \in B_k}\{d_j | c_j = C_k\} \tag{11}$$

Clearly, these parameters depend upon the sequence of the various jobs within block k. We define the block processing time P_k as the makespan of this sequence. Then $C_k = R_k + P_k$. Let σ be a feasible schedule with $K > 1$ blocks, and total tardiness $Z_1(\sigma) > 0$. Suppose it contains a job j such that $j \in \mathcal{B}_{[k]}$, $k > 1$, and $T_j > 0$. Let u be the job with the earliest release time in $B_{[k]}$, i.e., $R_{[k]} = r_u$. If j and all jobs in $B_{[k]}$ that start before j are left shifted such that u is now released at $C_{[k-1]}$, then T_j can be reduced because of the idle time between $B_{[k-1]}$ and $B_{[k]}$. Clearly, jobs that start after j remain unaffected by this left-shift and therefore, $Z_1(\sigma)$ is reduced. Note that all left-shifted jobs are merged with $B_{[k-1]}$. Such a left-shifting, with commensurate decrease in $Z_1(\sigma)$, can be done whenever a tardy job is found. Therefore, from induction, it follows that if $Z_1 > 0$, then all tardy jobs in any optimal solution must belong to $\mathcal{B}_{[1]}$. Furthermore, if $Z_1 > 0$, then $R_{[1]} = 0$ in an optimal solution

because otherwise, as before, Z_1 can be reduced by enforcing this condition. On the other hand, if $Z_1 = 0$, then clearly $\sum_{j \in \mathcal{B}_{[1]}} T_j = 0$. Therefore,

Remark 1. There exists an optimal schedule such that $Z_1 = \sum_{j \in \mathcal{B}_{[1]}} T_j$, and if $Z_1 > 0$, then $R_{[1]} = 0$.

Henceforth we consider only those schedules that satisfy the above property. For a given schedule σ, we define the set of *critical* jobs $\mathcal{J}_1$ as

$$\mathcal{J}_1 = \{j | j \in \mathcal{B}_{[1]}, \sum_{j \in \mathcal{B}_{[1]}} T_j > 0\}$$

Note that, while a tardy job is also a critical job, the reverse is not necessarily true. This is so because some jobs, which are in $\mathcal{B}_{[1]}$ and which are early in a given sequence, may become tardy in another sequence in which they are taken up with $B_{[k]}$, $k \geq 2$ because of the inserted idle times between blocks. Let $\mathcal{J}_2 = \mathcal{J} \backslash \mathcal{J}_1$ denote the set of *noncritical* jobs. Clearly, a job is noncritical if and only if it is early. Note that if $Z_1(\sigma) = 0$, then $\mathcal{J}_1$ is empty, and $\mathcal{J}_2$ must comprise at least one block. Alternatively, if $Z_1(\sigma) > 0$, then $\mathcal{J}_1$ consists of exactly one block, namely $B_{[1]}$; in this case, $\mathcal{J}_2$ can be empty.

Consider a block $B_{[k]} \in \mathcal{J}_2$ in σ. Because each job in $\mathcal{J}_2$ is early, $c_j \leq d_j$ for any $j \in \mathcal{B}_{[k]}$. Suppose that $c_j < d_j$ for all $j \in \mathcal{B}_{[k]}$. Let $\tau_j = d_j - c_j$ be the slack of j, and $u = arg\ min_{j \in \mathcal{B}_{[k]}}\{\tau_j\}$ be the job with the minimum slack in $B_{[k]}$. If $I_{k,k+1} = R_{[k+1]} - C_{[k]}$ is the idle time between $B_{[k]}$ and $B_{[k+1]}$, then, it is easy to see that all jobs in $\mathcal{B}_{[k]}$ can be right-shifted by $\tau = min\{\tau_u, I_{k,k+1}\}$ to yield a solution which has the same Z_1 as σ but a higher Z_2 value, and therefore, is superior to σ. If $\tau = \tau_u$, then u is completed exactly at its due date; otherwise, this right-shift results in the merger of block $B_{[k]}$ with $B_{[k+1]}$. Let the blocks be renumbered following this merger, and $u, \tau_u, I_{k,k+1}$, and τ be redefined for the modified block. If $c_j < d_j$ for all $j \in \mathcal{B}_{[k]}$, then similar right-shifting will result in a further increase in Z_2. This process will terminate if either $\tau = \tau_u$ in a merged block, or we reach the block sequenced last. But in the latter case, we again have $\tau = \tau_u$. Hence, in either case after the right shift, $c_u = d_u$. Repeating this argument whenever $c_j < d_j$ for all $j \in \mathcal{B}_{[k]}$, implies that

Remark 2. *In each block $B_k \in \mathcal{J}_2$ in an optimal schedule, there exists a pivotal job j_k such that $c_{j_k} = d_{j_k}$.*

$\mathcal{J}_1$ and $\mathcal{J}_2$ clearly depend on the schedule. On the other hand, if a partition $(\mathcal{J}_1, \mathcal{J}_2)$ that satisfies the condition given in Remark 1 is known then, as shown below, **ICP** separates into two problems – **ICP1** that considers only $\mathcal{J}_1$, and **ICP2** that considers only $\mathcal{J}_2$.

ICP1($\mathcal{J}_1$)

$$\text{Maximize } Z_2 = \sum_{j \in \mathcal{J}_1} r_j \tag{12}$$

subject to

$$\sum_{j \in \mathcal{J}_1} T_j = Z_1 = min_{\sigma \in \mathcal{S}}\{T(\sigma)\} \tag{13}$$

and

$$(3) - (7), \forall j \in \mathcal{J}_1.$$

ICP2($\mathcal{J}_2$)

$$Z_2 = \ max \sum_{j \in \mathcal{J}_2} r_j \tag{14}$$

subject to

$$T_j = 0, \ \forall j \in \mathcal{J}_2 \tag{15}$$

$$r_j - max_{j \in \mathcal{J}_1}\{c_j\} - 1 \geq 0 \ \forall j \in \mathcal{J}_2 \tag{16}$$

and,

$$(3) - (7), \forall j \in \mathcal{J}_2$$

where constraints (16) follow from (8). Together with (15), these constraints insure that the partition $(\mathcal{J}_1, \mathcal{J}_2)$ is feasible. Note that the dual objectives apply only to **ICP1**, while **ICP2** considers only the objective of maximizing the sum of job release times.

Let $J_2 = |\mathcal{J}_2|$ denote the number of noncritical jobs. With the introduction of blocks, **ICP2** can be reformulated as

ICP3($\mathcal{J}_2$)

$$Z_2 = \ max \sum_{k=1}^{J_2} \sum_{j \in B_{[k]}} r_j \tag{17}$$

subject to

$$\sum_{k=1}^{J_2} y_{jk} = 1, \ \forall j \in \mathcal{J}_2 \tag{18}$$

$$R_{[k+1]} - C_{[k]} - 1 \geq 0, \ \forall k \tag{19}$$

$$min_{j \in \mathcal{J}_2}\{r_j\} - max_{j \in \mathcal{J}_1}\{c_j\} - 1 \geq 0 \tag{20}$$

$$c_{uv} - c_{ji} \geq p_{uv} \vee c_{ji} - c_{uv} \geq p_{ji}, \ \forall O_{ji}, O_{uv} \in \mu_m, \forall j, u \in \mathcal{B}_k, \ \forall m, k \tag{21}$$

$$c_{j1} \geq r_j + p_{j1}, \ \forall j \in \mathcal{B}_k, \text{ and } \forall k \tag{22}$$

$$c_{jl} \geq c_{ji} + p_{jl}, \ \forall j \in \mathcal{B}_k, \ (i,l) \in \mathcal{A}_j, \text{ and } \forall k \tag{23}$$

$$c_{jN_j} \leq d_j, \forall j \in B_k, \forall k \tag{24}$$

$$c_{ji} \geq 0; \ r_j \geq 0, \ \forall i, \ \forall j \in \mathcal{B}_k, \ \forall k \tag{25}$$

where y_{jk} equals 1 if job j is assigned to block k, zero otherwise. Constraint (18) insures that each job is assigned to exactly one block, while constraints (19) and (20) follow from (8). Note that for a given partition of $\mathcal{J}_2$ into K blocks, $\{B_k\}_{k=1}^{K}$, that satisfies (18)-(20), **ICP3** decomposes into K independent subproblems. The kth subproblem, hereafter the *block maximum release time problem* (**BMRP**) is of the form

BMRP(k)

$$max \sum_{j \in B_{[k]}} r_j$$

subject to,

$$(21) - (25) \ \forall j \in \mathcal{B}_k.$$

4 Solution Approach

In view of the NP-completeness of **ICP**, problems of reasonable size are likely to be solved only by heuristic methods. We first give an outline of the suggested heuristic procedure; details of individual steps are discussed subsequently. The proposed heuristic procedure is now described; first, we give an outline of this method. Initially, we assume that all jobs can be finished on time, and set $\mathcal{J}_2 = \mathcal{J}; \mathcal{J}_1 = \emptyset$. We construct an initial partition of $\mathcal{J}_2$ into blocks, and attempt to solve BMRP for each block. If there is no feasible solution for a given block B_k (which implies that at least one job in B_k cannot be completed by its due date in this schedule), then all jobs in B_k are assigned to set $\mathcal{J}_1$, and $\mathcal{J}_2$ is updated. At the end of this step, we obtain a tentative partition of $\mathcal{J}$ into $\mathcal{J}_1$ and $\mathcal{J}_2$, and a partition of $\mathcal{J}_2$ into blocks that consist of early jobs when considered *independently*.

If $\mathcal{J}_1$ is not empty, we next solve **ICP1**, and assign $R_{[1]} = 0$. For the blocks in $\mathcal{J}_2$, we solve a relaxation of **ICP3** in which (19) and (20) are ignored. This is done by sequencing these blocks such that $c_{j_k} = d_{j_k}$ for each k. If the resulting schedule satisfies (19) and (20), then the algorithm terminates. It is easy to see that, with respect to the given sequence of jobs within $\mathcal{J}_1$ and within each $B_k \in \mathcal{J}_2$, this schedule is optimal.

Otherwise, suppose that (19) is violated such that $R_{[k+1]} < C_{[k]} + 1$, $k \geq 1$. In this case, block B_{k+1} is merged with block $B_{[k]}$, and **BMRP** is solved for $B_{[k]}$ with the enlarged set of jobs. The values of $c_{j_{[k]}}$ and $d_{j_{[k]}}$ are updated, and an attempt is made to sequence block $B_{[k]}$ such that $c_{j_{[k]}} = d_{j_{[k]}}$. This exercise is repeated for all such blocks that violate (19). On the other hand, if (20) is violated, then jobs in $B_{[2]}$ are merged with $\mathcal{J}_1$, and **ICP1** is re-solved for the modified $\mathcal{J}_1$. The indexes of blocks in $\mathcal{J}_2$ are updated accordingly, and the process is repeated until (20) is satisfied with the revised makespan of $\mathcal{J}_1$.

Because, each merger eliminates one block, this procedure will terminate within $|\mathcal{J}| - 1$ steps. A formal statement of the algorithm is given below.

Algorithm InputControl

Step 1: Initialization

Renumber all jobs as per EDD. Set $\mathcal{J}_1 = \emptyset$, $\mathcal{J}_2 = \mathcal{J}$. Determine the initial partition of $\mathcal{J}_2$ and the resulting set of job blocks $\{B_k\}$, $k = 1, \ldots, K$. Set $i = 0$.

Step 2: Sequencing Jobs within Blocks

a) Set $i \leftarrow i + 1$. If $i < K$, go to Step 2b. Else, go to Step 2d if $\mathcal{J}_1$ is not empty; go to Step 3, otherwise.

b) For block B_i, solve **BMRP**. If a feasible solution to **BMRP** is obtained, then determine P_i as the makespan of this sequence. Also determine the pivotal job for this block from

$$j_i = arg\ min_{j \in \mathcal{B}_i}\{d_j - c_j\}.$$

Record c_{j_i} and d_{j_i}, and go to Step 2a. Otherwise, if there is no feasible solution to **BMRP**, then go to Step 2c.

c) Set $\mathcal{J}_2 = \mathcal{J}_2 \backslash \mathcal{B}_i$, and $\mathcal{J}_1 = \mathcal{J}_1 \cup \mathcal{B}_i$. Set $i \leftarrow i - 1$, $K \leftarrow K - 1$, and go to Step 2a.

d) Solve **ICP1**. Determine $P(\mathcal{J}_1)$ as the makespan of the resulting schedule. Go to Step 3.

Step 3: Block Sequencing

a) If $\mathcal{J}_1$ is not empty, then set $\mathcal{B}_{[1]} = \mathcal{J}_1$, $R_{[1]} = 0$, and $C_{[1]} = P_{[1]} = P(\mathcal{J}_1)$. Sequence the remaining blocks in the nondecreasing order of $d_{j_{[k]}}$, and renumber blocks such that $k = [k]$, $k = 1, \ldots, K$. If $\mathcal{J}_1$ is not empty, set $i = 2$; otherwise, set $i = 1$.

b) Schedule block B_i such that $c_{j_i} = d_{j_i}$. Determine C_i and R_i in the resulting schedule. If $R_i \geq C_{i-1} + 1$, go to Step 3c. Else, go to Step 4.

c) If $i = K$, stop. Otherwise, set $i \leftarrow i + 1$ and go to Step 3b.

Step 4: Block Merging

Merge B_i with B_{i-1}. Set $B_{k-1} \leftarrow B_k$, for $i \leq k \leq K$. If $i = 2$, and $\mathcal{J}_1$ is not empty, go to Step 2d. Otherwise, solve **BMRP** for the enlarged block B_{i-1}. Update P_{i-1}, j_{i-1}, $c_{j_{i-1}}$ and $d_{j_{i-1}}$. Go to Step 3a.

The major steps in the algorithm are i) the determination of the initial blocks, ii) solving **BMRP**, and iii) solving **ICP1**. These are now discussed.

4.1 Determination of Initial Blocks

Clearly, each job can be considered as an independent block initially. However, the overall computational effort can generally be reduced by identifying subsets of jobs that are likely to be in same block in an optimal sequence. In this procedure, we group jobs with *overlapped processing* to form the initial set of blocks. A pair $u, v \in \mathcal{J}$ of jobs are said to have overlapped processing if they satisfy the property that if $d_v \geq d_u$, then $d_v - p_v < d_u$.

Consider a pair (u, v) of jobs that satisfy this property. Suppose that $T_u > 0$ in an optimal schedule. Then $u \in \mathcal{J}_1$. If $v \notin \mathcal{J}_1$, then

$$c_v \geq r_v + p_v > c_u + p_v > d_u + p_v > d_v$$

and v is tardy as well. But this contradicts Remark 1; hence v must also lie in $\mathcal{J}_1$. It can similarly be shown that in an optimal schedule, if $T_v > 0$, then $u, v \in \mathcal{J}_1$. Therefore, if a pair of jobs have the overlapped processing property, then they belong to the same block if any one of them is tardy. Counterexamples can, however, be easily constructed to show that this result does not hold if both jobs are early. Nonetheless, preliminary computational experiments revealed that, in most of the final sequences obtained, jobs with overlapped processing are produced in the same block.

An $O(N^2)$ algorithm for identifying pairs of jobs with the overlapped processing property is given below. Recall that all jobs are numbered in the nondecreasing order of their due dates.

Algorithm InitialBlocks

Step 1. Set $K = 1$; $j = N$; and $k = K$.

Step 2. Set $y_{jk} = 1$; and $j = j - 1$. If $j = 0$, go to Step 4. Else go to Step 3.

Step 3. If there exists a block B_k, $k = 1, 2, \ldots, K$ such that $d_j \geq d_u - p_u$, $\forall u \in \mathcal{B}_k$, then go to Step 2. Else, set $K \leftarrow K + 1$; $k = K$, and go to Step 2.

Step 4. Renumber blocks: Set $k \leftarrow K + 1 - k$, $k = 1, 2, \ldots, K$.

It is easy to verify that at most one B_k will satisfy the condition stated in Step 3.

4.2 The Block Maximum Release Time Problem

The problem of sequencing jobs in each block such that the sum of their release times is maximized subject to their completion on or before their due dates is easily shown to be NP-complete. We propose an iterative, improvement heuristic method for solving this problem. We first discuss the procedure for generating the initial solution, and subsequently describe the improvement step.

4.2.1 Initial Solution

We construct the initial schedule for **BMRP** by considering its inverse problem – the completion time problem **CTP**. For a given instance of **BMRP**, the corresponding instance of **CTP** is constructed by reversing the order of operations in each job such that $p'_{ji} = p_{j,N_j+1-i}$, where the "prime" distinguishes the parameters and variables in **CTP**. Each job $j \in \mathcal{J}$ is assigned a ready time $q'_j = d_{max} - d_j$, where d_{max} is the maximum of the due dates of jobs within the given block. For block B_k, **CTP** is stated as

CTP(k)

$$min \sum_{j \in B_k} c'_j$$

subject to,

$$c'_{uv} - c'_{ji} \geq p'_{uv} \vee c'_{ji} - c'_{uv} \geq p'_{ji}, \quad \forall\, O'_{ji}, O'_{uv} \in \mu'_m, \forall j, u \in \mathcal{B}_k, \quad \forall m$$

$$c'_{j1} \geq q'_j + p'_{j1}, \quad \forall j \in \mathcal{B}_k$$

$$c'_{jl} \geq c'_{ji} + p'_{jl}, \quad (i,l) \in \mathcal{A}'_j, \quad \forall j \in \mathcal{B}_k$$

$$c'_{ji} \geq 0, \quad \forall j \in \mathcal{B}_k$$

Any schedule feasible to **BMRP** has an equivalent, antithetic schedule feasible to **CTP** with operation completion times $c'_{ji} = d_{max} - s_{j,N_j+1-i}$, and job completion times

$$c'_j = d_{max} - r_j. \tag{26}$$

The equivalence between these two problems is illustrated in the 3-job, 3-machine example shown in Figure 2. In this example, we have $d_2 < d_3 < d_1 = d_{max}$ for **BMRP**. The job ready times in the corresponding instance of **CTP** are $q'_1 = 0$, $q'_2 = d_1 - d_2$, and $q'_3 = d_1 - d_3$.

INSERT FIGURE 2 HERE

We note here that Ahmadi and Bagchi (1992) use a similar problem inversion for addressing a nonregular scheduling objective. They refer to **CTP** as the mirror image problem, and independently derive a formal equivalence between **CTP** and the minimum job idleness problem in a flow shop.

BMRP is then solved by first solving **CTP** to determine job completion times c'_1, c'_2, and c'_3, and then using (26) to obtain the ready times r_j. **CTP** remains a hard problem even for a single machine system (Hariri and Potts 1983). We generate a *nondelay* schedule (Baker 1974, page 191) in which ties between competing jobs are broken in favor of the job with the shortest imminent operation time. The order of operations generated for **CTP** is reversed to obtain the desired sequence of jobs within the block for **BMRP**. Let c_{ji} and s_{ji}, respectively, denote the completion time and the start time of operation i within job j in this sequence.

4.2.2 Schedule Improvement

At the start of the improvement phase, the machines are ranked in the nonincreasing order of their workloads. We consider one machine at a time in the ranked order starting with the machine at the top of the list. For each machine, an attempt is made to reschedule operations, and the revised schedule is then used to update the parameters considered for the next machine. When all machines have been considered, the first machine on the list is reinvestigated at the beginning of the next cycle. This iterative procedure terminates when no improvement in the total release times for all jobs in the block is obtained in one complete cycle.

The problem to be solved at each machine requires maximizing the sum of release times of operations subject to their availability by their ready times and completion by the specified due dates. For operation i in job j, the ready time is

$$q_{ji} = \begin{cases} c_{j,i-1} & \text{if } i > 1 \\ 0 & \text{otherwise} \end{cases}$$

and the due date is

$$d_{ji} = \begin{cases} s_{j,i+1} & \text{if } i < N_j \\ d_j & \text{otherwise.} \end{cases}$$

This problem is equivalent to minimizing total earliness subject to specified ready times (**ERP**). Let $\mathcal{I}$ be the set of operations processed on the machine currently under consideration. For the ease of presentation, we introduce the following modified notation. Let a_i, b_i, t_i, and f_i denote, respectively, the ready time, due date, processing time, and completion time of operation i. Let $\mathcal{I}_j$ be the set of adjacent operations in job j. **ERP** can then be stated as:

$$\text{Minimize } \sum_{i \in \mathcal{I}} (b_i - f_i)$$

subject to,

$$f_i \geq a_i + t_i, \quad \forall i$$

$$f_i \leq b_i, \quad \forall i$$

$$f_i - f_h \geq t_i \vee f_h - f_i \geq t_h \quad \forall i, h, i \neq h$$

$$f_l \geq f_i + t_l, \quad \forall (i,l) \in \mathcal{I}_j, \forall j$$

$$f_i \geq 0, \quad \forall i$$

Because the regular earliness problem is NP-complete, **ERP** is NP-complete as well; therefore, it is unlikely that a polynomial time algorithm exists for solving it exactly. We construct a heuristic solution algorithm which is a modification of the approximation method suggested by Chand and Schneeberger (1985) for the regular earliness problem. For the sake of completeness, we give an outline of the Chand and Schneeberger heuristic below.

The algorithm uses the Smith (1956) heuristic to build the schedule backwards by assigning one operation at a time. Suppose that at the end of a step i operations have been scheduled in positions $N - i + 1$ through N. At the next iterative step, from among all operations feasible for assignment, the operation with the smallest processing time is selected and scheduled

such that it is completed at $\tau = min\{b_i, f_{[N-i+1]} - t_{[N-i+1]}\}$. The schedule developed in this manner is optimal if $t_{[i]} \geq t_{[i+1]}$, $i = 1, \ldots, I-1$.

Suppose $\sigma^1 = \{i, \sigma\}$ and $\sigma^2 = \{h, i, \sigma\}$ are two subsequences that satisfy the above optimality condition. Suppose further that the operation g selected at the next step yields the subsequence $\sigma^3 = \{g, h, i, \sigma\}$ that violates this condition. The heuristic then backtracks to the partial solution σ^1 to consider two alternative completions. In the first sequence, g is assigned to the position just ahead of i and the rest of the schedule is developed by using the Smith method. In the second completion, g is assigned to the next to last position among the unscheduled jobs and, subject to this assignment, the best sequence obtained by using the Smith method is found. The better of these two completions is selected.

An important step in the above approach is to ascertain the feasibility of assigning an operation to a given position. When all operations are available at time zero, the feasibility of a partial solution can be verified by sequencing the unscheduled operations in an Earliest Due Date (EDD) order. If the maximum lateness L_{max} for the resulting sequence is nonpositive, then there is at least one feasible completion of the partial solution; otherwise, the partial solution is infeasible. For nonzero ready times, verifying the feasibility of a partial schedule to **ERP** requires solving the problem of minimizing the maximum lateness of the unscheduled operations subject to ready times ($I/1/r_i \geq 0/L_{max}$). As before, a feasible completion exists if and only if the optimal $L_{max} \leq 0$. Although $I/1/r_i \geq 0/L_{max}$ is NP-complete in the strong sense, effective algorithms for solving it have been proposed by McMahon and Florian (1975) and Carlier (1982) that are based on implicit enumeration. We modify Carlier's algorithm in order to improve its efficiency in the context of **ERP**. This is done by identifying conditions that lead to infeasible completions. Clearly, a positive lower bound on L_{max} gives one such condition. Our computational experience revealed that the overall solution time improves if preprocessing to insure that two additional conditions are satisfied is carried out. These conditions are: 1) $t \geq a_i + t_i$, for all $i \in \mathcal{U}$, and 2) $t \geq \sum_{i \in \mathcal{U}} t_i$, where t is the start time of the most recently scheduled operation and $\mathcal{U}$ is the set of operations yet to be scheduled.

Modifying the start and the completion times of an operation of any job on a given machine results in revising the ready time for the succeeding operation and the due date for the

preceding operation in the same job. Thus, in general, rescheduling a machine leads to updating the parameters for the next machine to be considered. Although it is difficult to theoretically guarantee convergence of the objective function value, we found it to be so in our computational experience; indeed, the bulk of the improvement was obtained within the first three cycles.

In some cases, the solution to **BMRP** results in decomposing a block into two or more blocks. In such cases, the total number of blocks is updated.

4.3 Problem ICP1

As discussed earlier, **ICP1** is bicriterion problem with the primary objective of minimizing mean tardiness of all jobs in $\mathcal{J}_1$. Our solution method consists of solving the mean tardiness problem first. The job due dates are then revised, if necessary, based on the resulting schedule. Given these due dates, the maximum release time problem is next solved following the procedure discussed in the previous section.

In view of the strong NP-completeness of the mean tardiness problem, we propose using the heuristic method developed by Raman, Talbot and Rachamadugu (1989) which is briefly described now. This method generates an initial solution by first setting the operation due dates (ODDs) loosely at the maximum values that they can assume without delaying the corresponding jobs. The due date of operation i in job j is then given by

$$d_{ji} = \begin{cases} d_j, & \text{if } i = N_j \\ d_{j,i+1} - p_{ji}, & \text{otherwise} \end{cases}$$

The sequence of operations at each machine is obtained by applying the Modified Operation Due Date (See, for example, Baker 1984b) rule. Each machine is then considered in the order of its relative workload, and an attempt is made to revise the schedule of operations on that machine by modifying their ODDs. The jobs processed on this machine are ranked in the nonincreasing order of their tardiness. For any operation in a given job with positive tardiness, first we determine the appropriate interval for searching for its ODD. For each ODD value in this interval, the entire system is rescheduled. The value that yields the

minimum total tardiness is returned as the ODD for that operation. This step is repeated for all other operations of that job processed on the machine under consideration, for all other tardy jobs on that machine following their rank order, and for all machines in the system.

From the minimum mean tardiness schedule, job due dates are revised according to $d_j \leftarrow max(d_j, c_j)$. Subject to these due dates, we next solve **BMRP** for the jobs in $\mathcal{J}_1$. Because a feasible schedule is available initially, we can go directly to the schedule improvement phase discussed in §4.2.2.

5 Computational Experience

Three sets of experiments were conducted. The first two sets evaluated the performance of algorithm **InputControl**, while the third set addressed the improvement obtained in the sum of job release times by solving the dual-objective input control problem over the single-objective mean tardiness problem. While it is clear that explicit consideration of the secondary objective should result in better solutions, the purpose of this set of experiments was to examine the margin of possible improvement achieved by using algorithm **InputControl**.

5.1 Experimental Design

The first set of experiments considered a single-machine problem with 10 jobs. Job processing times varied uniformly in the interval (0,100). The due date of d_j of a given job j was determined by

$$d_j = F_j(\sum_j p_j) \tag{27}$$

where F_j was sampled from a uniform distribution in the interval $(\overline{F} - R\overline{F}/2, \overline{F} + R\overline{F}/2)$. $\overline{F}$ and R respectively control the tightness and the variability of job due dates. Eight levels of $\overline{F}$ – 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50, and 0.60, were used to give a wide range of due date tightness. At each of these eight levels, two values of R – 0.5, and 1.5, were used to provide a total of 16 combinations of due date tightness and variability.

Ten instances of each problem scenario were randomly generated. We used IBM's Optimization Subroutine Library (OSL) to determine the optimal solution under each instance. The solution values with respect to total tardiness and the sum of job release times obtained under both the optimal and the proposed solution approaches were recorded and averaged over the ten problem instances. For reporting purposes, the total tardiness value was divided by the sum of job processing times to obtain the normalized value. Similarly, the sum of job release times was normalized with respect to the sum of job due dates. The results are shown in Table 1 in which Z_1 and Z_2, respectively, denote normalized total tardiness and normalized sum of release times. In total, the first set of experiments considered 160 problems.

The second set evaluated the performance of **InputControl** with respect to known upper bounds for a 2-machine flow shop problem in which all job due dates are equal. Ahmadi and Bagchi (1992) develop an approach based on Lagrangean relaxation for deriving an upper bound on the solution value for this problem. This approach dualizes constraints (5) with nonnegative multipliers; this decomposes the original problem into two subproblems, one for each machine. The individual subproblems require maximizing the sum of weighted job completion times subject to no jobs being tardy. While this problem is NP-complete, Ahmadi and Bagchi construct a multiplier adjustment procedure for generating an upper bound on its solution value. The Lagrange multipliers are determined through a modified subgradient optimization procedure. To our knowledge, this is the only other upper bound available currently for multiple machine problems that address similar objectives. Let UB denote this upper bound. [Ahmadi and Bagchi also derive an upper bound for *permutation schedules* when job due dates are distinct. We do not consider this case here because **InputControl** is likely to generate many nonpermutation schedules as well.]

We considered two problem sizes; the smaller problem considered 25 jobs while the larger problem had 50 jobs. The operation processing times were selected from a uniform distribution in the interval (0,100). All jobs had the same due date d which was determined by

$$d = \overline{F}(\sum_j p_j).$$

Five values of $\overline{F}$ – 0.6, 0.7, 0.8, 0.9, and 1.0 were used to generate increasingly loose due dates. [As reported in Ahmadi and Bagchi's (1992) study as well, we found that $\overline{F} < 0.6$ led to due date infeasibility in many cases.] As in the first set of experiments, ten problem instances were solved for each scenario. For each instance, the ratio of Z_2 obtained from **InputControl** to UB was recorded. The results reported in Table 2 indicate the average as well as the minimum and the maximum values of these ratios over the ten problem instances. In all, the second set of experiments considered 100 problems.

The third set addressed a 5-machine job shop for three values of N – 10, 20, and 30. Each job was assigned 5 operations, and the machine visitation sequence was assigned randomly though successive operations of a given job were processed in different machines. Ten randomly generated instances of each problem scenario were solved using two approaches. The first approach employed **InputControl** that considered both primary and secondary objectives. The second approach considered only the primary objective of minimizing total tardiness. In order to restrict the computational costs within reasonable limits, the Modified Operation Due Date (MOD) rule [see, for example, Baker 1984] was used for solving the mean tardiness problem under both approaches. Operation processing times were selected from a uniform distribution in the interval (0,100). As in the first set, jobs due dates were determined by (27) and 16 combinations of $\overline{F}$ and R were used. Table 3 gives the normalized values of Z_1 and Z_2 under the two approaches, as well as the ratio $\rho = Z_2(\textbf{InputControl})/Z_2(\text{MOD})$. We also report the computation times for **InputControl** on an IBM 3081-GX2 computer. In total, the third set of experiments considered 480 problems.

5.2 Experimental Results

It can be seen from Table 1 that **InputControl** finds solution values that are close to the optimal values, in particular when $R = 1.5$. Table 2 indicates that its performance extends to two-machine flow shops. On average, its solution value is within 6.9% of the upper bound across the tested problems.

As indicated in Table 3, Z_1 decreases in all cases as due dates become progressively looser with an increase in $\overline{F}$. For MOD, there is a decrease in Z_2 as well. Recall that Z_2 is normalized with respect to the sum of due dates; while an increase in $\overline{F}$ increases the due dates, the release times do not change appreciably. On the other hand, for **InputControl**, after an initial decrease, Z_2 starts increasing as the release times are extended commensurate with the due dates. Note that for the same value of $\overline{F}$, Z_2 is generally higher for higher values of R. This is due to the fact that, with greater dispersion in job due dates, the final schedule contains a larger number of blocks. Hence, a larger portion of jobs is completed close to their due dates with consequent increase in release times as well.

As indicated by the ρ values in Table 3, **InputControl** is seen to yield substantial improvement in Z_2 values over MOD even when due dates are quite tight. This improvement increases as due dates become looser, although for $R = 0.5$, there is a small decrease in ρ initially at small $\overline{F}$ values. Note also that **InputControl** gives marginally better total tardiness values as well. This is due to the fact that **InputControl** first schedules jobs in $\mathcal{J}_1$, while MOD considers all jobs to be schedulable at any given time. Therefore, in constructing a nondelay schedule, MOD frequently takes up a nonurgent job for processing if by doing so, machine idleness is avoided. This could delay the processing of an urgent job that arrives at the machine soon thereafter.

6 SUMMARY

This paper examines the effectiveness of employing input control as a mechanism for deferring job release in a just-in-time system. The input control decision is modeled as a dual objective problem of minimizing total job tardiness and maximizing the sum of job release times in a lexicographic manner.

In view of the problem complexity, a heuristic solution method is constructed in which jobs are grouped into blocks for simultaneous processing. The membership of each block is updated in an iterative manner by solving a sequence of subproblems. Our computational experience indicates that this approach yields substantial improvement in job release times,

without adversely affecting the tardiness values, over an approach that considers the single objective of minimizing total tardiness. This result is of interest to an operating manager who is responsible for controlling work-in-process and finished goods inventory related costs in addition to meeting due dates effectively.

Acknowledgements

The author thanks the Department Editor and two anonymous referees for their thoughtful and detailed comments, and suggestions that have greatly improved this paper.

REFERENCES

1. Ahmadi, R. H. and U. Bagchi (1992), "Minimizing Job Idleness in Deadline Constrained Environments," *Operations Research*, Vol. 40, 972–985.

2. Baker, K. R. (1974), *Introduction to Sequencing and Scheduling*, John Wiley and Sons, New York, NY.

3. Baker, K. R. (1984a), "The Effects of Input Control in a Simple Scheduling Model," *Journal of Operations Management*, Vol. 4, 99–112.

4. Baker K. R. (1984b), "Sequencing Rules and Due date Assignments in a Job Shop," *Management Science*, Vol. 30, 1093–1104.

5. Baker, K. R. and G. D. Scudder (1990), "Sequencing with Earliness and Tardiness Penalties: A Review," *Operations Research*, Vol. 38, 22–36.

6. Carlier, J. (1982), "The One-Machine Scheduling Problem," *European Journal of Operational Research*, Vol. 11, 42–47.

7. Chand, S. and H. Schneeberger (1985), "A Two-Stage Approximation Method for the Single Machine Weighted Earliness Problem," Working Paper # 424, Graduate School of Business Administration, The University of Michigan, Ann Arbor, MI.

8. Chand, S. and H. Schneeberger (1986), "A Note on the Single Machine Scheduling Problem with Minimum Weighted Completion Time and Maximum Allowable Tardiness," *Naval Research Logistics Quarterly*, Vol. 33, 551–557.

9. Du, Z. and J. Y.-T. Leung (1990), "Minimizing Total Tardiness on One-Machine is NP-Hard," *Mathematics of Operations Research*, Vol. 15, 483–495.

10. Fry, T. D., R. D. Armstrong and J. H. Blackstone (1987), "Minimizing Weighted Absolute Deviation in Single Machine Scheduling," *IIE Transactions*, Vol. 19, 445–450.

11. Garey, M. R., R. E. Tarjan and G. T. Wilfong (1988), "One- Processor Scheduling with Symmetric Earliness and Tardiness Penalties," *Mathematics of Operations Research*, Vol. 13, 330–348.

12. Hariri, A. M. A. and C. N. Potts (1983), "An Algorithm for Single Machine Sequencing with Release Dates to Minimize To Weighted Completion Time," *Discrete Applied Mathematics*, Vol. 5, 99–109.

13. Irastorza, J. C. and R. H. Deanne (1974), "A Loading and Balancing Methodology for Job Shop Control," *AIIE Transactions*, Vol. 6, 302–307.

14. Kim, Y.-D. and C. A. Yano (1986), "Algorithms for Single Machine Scheduling Problems Minimizing Tardiness and Earliness," Technical Report 86–40, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI.

15. McMahon, G. and M. Florian (1975), "On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness," *Operations Research*, Vol. 23, 475–482.

16. Panwalker, S. S., R. A. Dudek and M. L. Smith (1973), "Sequencing Research and the Industrial Problem," in *Symposium on the Theory of Scheduling*, edited by S. E. Elmaghraby, Springer-Verlag, Berlin.

17. Raman, N., F. B. Talbot and R. V. Rachamadugu (1989), "Scheduling a General Flexible Manufacturing System to Minimize Tardiness Related Costs," BEBR Faculty Working Paper # 89-1547, Univ. of Illinois at Urbana-Champaign, Champaign, IL.

18. Rinnooy Kan, A. H. G. (1976), *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague, Netherlands.

19. Shimoyashiro, S., K. Isoda and H. Awane (1984), "Input Scheduling and Load Balance Control for a Job Shop," *International Journal of Production Research*, Vol. 22, 597–605.

20. Smith, M. L., R. Ramesh, R. A. Dudek and E. L. Blair (1986),"Characteristics of U.S. Flexible Manufacturing Systems – A Survey," in *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, MI, 477–486.

21. Smith, W. E. (1956), "Various Optimizers for Single Stage Production," *Naval Research Logistics Quarterly*, Vol. 3, 56–66.

22. Stecke, K. E. (1983), "Formulation and Solution of Nonlinear Integer Programming Problems for Flexible Manufacturing Systems," *Management Science*, Vol. 29, 273–288.

23. Vepsalainen, A. P. J. and T. E. Morton (1987), "Priority Rules for Job Shops with Weighted Tardiness Costs," *Management Science*, Vol. 33, 1035–1047.

24. Yano, C. A. and Y.-D. Kim (1991), "Algorithms for a Class of Single-Machine Weighted Tardiness and Earliness Problems," *European Journal of Operational Research*, Vol. 52, 167–178.

TABLE 1

Performance of **InputControl**: The Single-Machine Case

R	$\overline{F}$	**InputControl**		Optimal	
		Z_1	Z_2	Z_1	Z_2
0.5	0.05	3.42	5.88	3.42	5.88
	0.10	3.08	3.06	3.07	3.05
	0.15	2.79	2.14	2.79	2.14
	0.20	2.42	1.66	2.42	1.66
	0.25	2.26	1.47	2.23	1.44
	0.30	1.76	1.16	1.76	1.16
	0.40	1.34	0.97	1.33	0.92
	0.50	0.96	0.85	0.95	0.81
1.5	0.05	3.59	6.47	3.59	6.47
	0.10	3.08	3.07	3.08	3.07
	0.15	2.65	2.09	2.65	2.09
	0.20	2.45	1.75	2.45	1.75
	0.25	2.05	1.38	2.05	1.48
	0.30	1.90	1.28	1.90	1.28
	0.40	1.07	0.96	1.07	0.96
	0.50	0.68	0.86	0.68	0.86

TABLE 2

Performance of **InputControl**: The Two-Machine Flow Shop Case

N	$\overline{F}$	Z_2(**InputControl**)$/UB$		
		Minimum	*Average*	Maximum
25	0.6	0.905	0.944	0.967
	0.7	0.912	0.949	0.996
	0.8	0.917	0.935	0.978
	0.9	0.922	0.959	0.990
	1.0	0.935	0.963	0.993
50	0.6	0.914	0.934	0.967
	0.7	0.904	0.931	0.961
	0.8	0.925	0.940	0.973
	0.9	0.933	0.942	0.964
	1.0	0.936	0.951	0.969

TABLE 3

Comparison of **InputControl** with MOD: The Job Shop Case

N	R	$\overline{F}$	*MOD*		**InputControl**		ρ	*CPU*
			Z_1	Z_2	Z_1	Z_2		*seconds*
10	0.5	0.05	1.426	0.633	1.424	1.065	1.682	0.074
		0.10	0.876	0.296	0.862	0.458	1.547	0.078
		0.15	0.728	0.370	0.724	0.519	1.402	0.114
		0.20	0.319	0.236	0.312	0.387	1.640	0.119
		0.25	0.115	0.188	0.114	0.427	2.272	0.098
		0.30	0.018	0.162	0.018	0.503	3.105	0.127
		0.40	0.009	0.166	0.009	0.616	3.711	0.137
		0.50	0.000	0.113	0.000	0.693	6.133	0.016
	1.5	0.05	1.303	0.876	1.299	1.085	1.238	0.068
		0.10	0.860	0.312	0.858	0.517	1.657	0.080
		0.15	0.510	0.236	0.507	0.431	1.826	0.097
		0.20	0.415	0.261	0.415	0.514	1.969	0.116
		0.25	0.144	0.181	0.142	0.488	2.696	0.123
		0.30	0.123	0.148	0.121	0.597	4.034	0.123
		0.40	0.062	0.134	0.062	0.668	4.985	0.111
		0.50	0.034	0.109	0.033	0.729	6.688	0.142

TABLE 3 (continued)

Comparison of **InputControl** with MOD: The Job Shop Case

N	R	$\overline{F}$	*MOD*		**InputControl**		ρ	*CPU*
			Z_1	Z_2	Z_1	Z_2		*seconds*
20	0.5	0.05	1.819	0.838	1.802	1.208	1.441	0.941
		0.10	1.242	0.533	1.224	0.708	1.328	0.906
		0.15	0.534	0.387	0.525	0.549	1.419	0.866
		0.20	0.102	0.292	0.102	0.497	1.702	1.502
		0.25	0.013	0.245	0.012	0.549	2.241	1.586
		0.30	0.000	0.223	0.000	0.654	2,933	1.434
		0.40	0.000	0.150	0.000	0.768	5.120	0.977
		0.50	0.000	0.132	0.000	0.819	6.204	2.094
	1.5	0.05	1.812	0.920	1.795	1.247	0.491	1.355
		0.10	1.122	0.504	1.097	0.723	1.434	1.028
		0.15	0.610	0.443	0.595	0.655	1.478	0.885
		0.20	0.174	0.309	0.171	0.640	2.071	1.109
		0.25	0.067	0.263	0.065	0.696	2.646	1.404
		0.30	0.008	0.233	0.008	0.763	3.275	1.364
		0.40	0.005	0.157	0.005	0.809	5.142	1.091
		0.50	0.000	0.131	0.000	0.875	6.679	1.232

TABLE 3 (continued)

Comparison of **InputControl** with MOD: The Job Shop Case

N	R	$\overline{F}$	*MOD*		**InputControl**		ρ	*CPU*
			Z_1	Z_2	Z_1	Z_2		*seconds*
30	0.5	0.05	2.391	0.970	2.346	1.285	1.325	4.412
		0.10	1.545	0.590	1.507	0.791	1.341	6.970
		0.15	0.600	0.462	0.578	0.626	1.355	5.955
		0.20	0.092	0.335	0.090	0.544	1.624	8.149
		0.25	0.000	0.282	0.000	0.668	2.368	9.060
		0.30	0.000	0.239	0.000	0.740	3.096	12.636
		0.40	0.000	0.178	0.000	0.764	4.341	10.453
		0.50	0.000	0.137	0.000	0.808	5.898	15.654
	1.5	0.05	2.381	0.947	2.343	1.348	1.423	4.534
		0.10	1.134	0.628	1.102	0.754	1.201	4.399
		0.15	0.467	0.484	0.448	0.684	1.413	3.530
		0.20	0.082	0.337	0.080	0.734	2.178	8.314
		0.25	0.054	0.306	0.049	0.717	2.343	9.642
		0.30	0.000	0.237	0.000	0.731	3.084	13.711
		0.40	0.000	0.178	0.000	0.842	4.730	20.451
		0.50	0.000	0.154	0.000	0.767	4.980	26.066

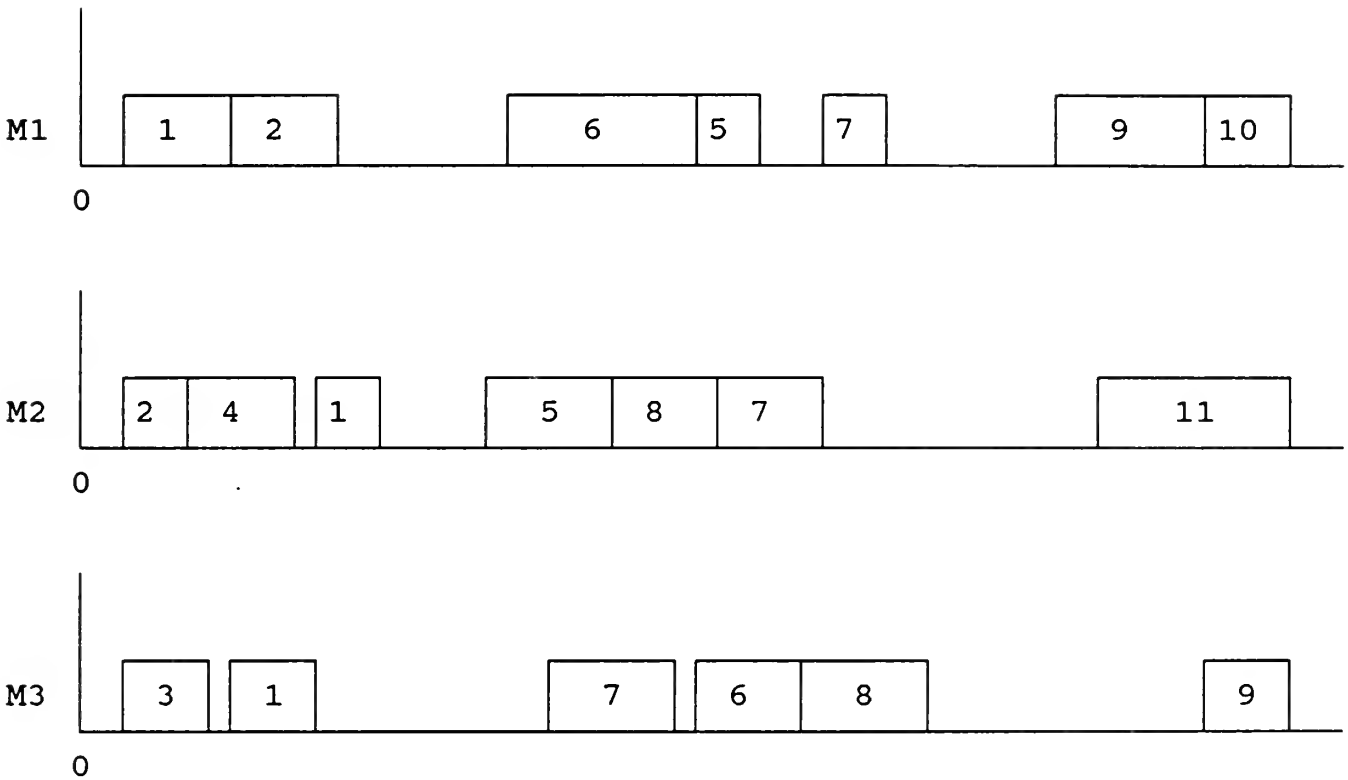

Figure 1: A Typical Feasible Schedule

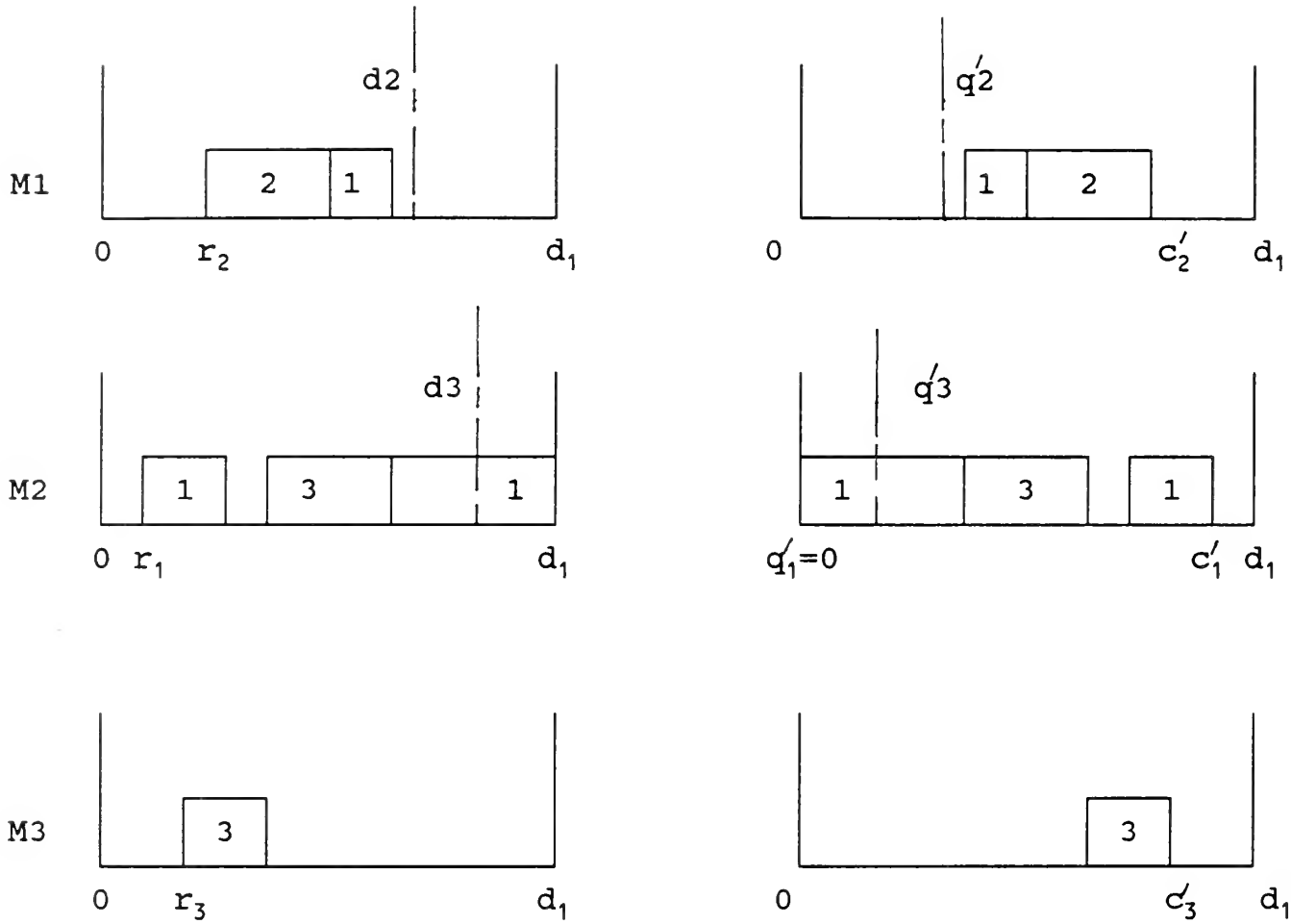

Figure 2: An Illustration of the Equivalence between **BMRP** and **CTP**